

COMPUTER SYSTEM

Publication number: JP2001175486

Publication date: 2001-06-29

Inventor: SATO MASAHIDE; ARAI TOSHIAKI

Applicant: HITACHI LTD

Classification:

- International: G06F12/14; G06F9/46; G06F21/24; G06F12/14;
G06F9/46; G06F21/00; (IPC1-7): G06F9/46; G06F12/14

- European:

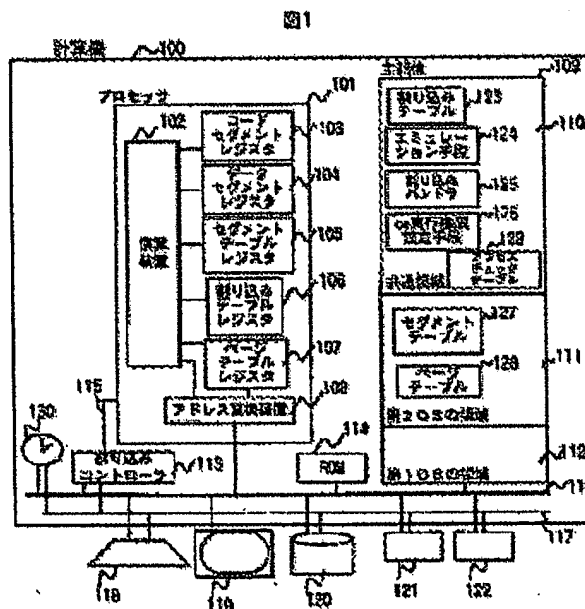
Application number: JP19990363154 19991221

Priority number(s): JP19990363154 19991221

Report a data error here

Abstract of JP2001175486

PROBLEM TO BE SOLVED: To provide a computer system for realizing inter-OS resource protection, without providing inter-OS resource protecting mechanism which is constituted of special hardware by allowing plural OS to coexist. **SOLUTION:** An OS which normally operates with the highest execution authority is allowed to operate with the reduced execution authority level, so that a resource can be protected between OS, and a processor is set with an exception can be generated, when the OS performs access to the resource to be protected. Thus, the processor is informed that the OS which is allowed to operate with the reduced execution authority which tries to make access to the resource to be protected. An interrupt table for each OS is set so that emulation processing when the execution is notified can be registered, and the emulation processing to be executed when the exception is notified to the interrupt table of the OS, which is allowed to operate with the reduced execution authority is generated. Thus, before the OS which is allowed to operate with the reduced execution authority performs access to the resource to be protected, the emulation of the access can be attained by checking whether the access is really available, so that the access to the resource to be protected can be restricted.



Data supplied from the esp@cenet database - Worldwide

(19) 日本国特許庁 (J P)

(12) 公開特許公報 (A)

(11) 特許出願公開番号

特開2001-175486

(P2001-175486A)

(43) 公開日 平成13年6月29日 (2001.6.29)

(51) Int.Cl. ⁷	識別記号	F I	テマコード* (参考)
G 0 6 F 9/46	3 5 0	G 0 6 F 9/46	3 5 0 5 B 0 1 7
12/14	3 1 0	12/14	3 1 0 L 5 B 0 9 8

審査請求 未請求 請求項の数 4 O L (全 14 頁)

(21) 出願番号 特願平11-363154

(22) 出願日 平成11年12月21日 (1999.12.21)

(71) 出願人 000005108

株式会社日立製作所

東京都千代田区神田駿河台四丁目6番地

(72) 発明者 佐藤 雅英

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(72) 発明者 新井 利明

神奈川県川崎市麻生区王禅寺1099番地 株

式会社日立製作所システム開発研究所内

(74) 代理人 100078134

弁理士 武 顕次郎

Fターム(参考) 5B017 AA01 BA03 BB06 CA01

5B098 BA04 BB05 CC01 GA02 GD02

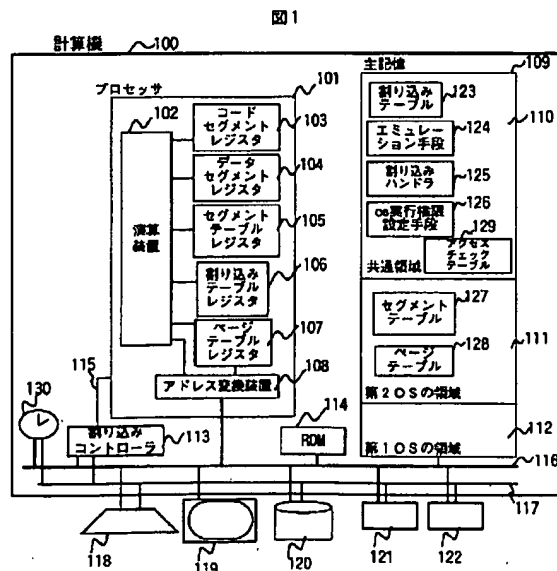
HH07

(54) 【発明の名称】 計算機システム

(57) 【要約】

【課題】 複数OSを共存させ、特別なハードウェアによるOS間資源保護機構なしに、OS間の資源保護を実現することのできる計算機システムを得る。

【解決手段】 OS間で資源を保護するために、通常、最高の実行権限を有して動作するOSの実行権限レベルを下げて動作させ、保護すべき資源にアクセスした場合に例外が発生するようにプロセッサを設定する。これにより、実行権限を下げて動作させているOSが、保護すべき資源にアクセスしようとしたことがプロセッサに通知される。例外が通知された場合のエミュレーション処理を登録するために、OS毎の割り込みテーブルを設置し、実行権限を下げて動作させるOSの割り込みテーブルに例外が通知された場合に実行すべきエミュレーション処理を登録する。これにより、実行権限を下げて動作させているOSが保護すべき資源にアクセスする前に、本当にアクセスして良いかチェックしてからアクセスをエミュレーションするため、保護すべき資源へのアクセスを制限できる。



【特許請求の範囲】

【請求項1】 計算機資源のアクセス権を制限可能な実行権限レベルを複数有し、1つの計算機上で、複数のオペレーティングシステムが独自のアドレス空間と、オペレーティングシステム間共通のアドレス空間とを有し、プロセッサを複数のオペレーティングシステムにより共有して動作する計算機システムにおいて、オペレーティングシステムが実施する処理を代替するエミュレーション手段と、オペレーティングシステムの実行権限レベルを変更するOS実行権限設定手段と、オペレーティングシステムの計算機資源へのアクセス可否情報が設定されているアクセスチェックテーブルと、オペレーティングシステムが計算機資源にアクセスしようとするときにアクセスチェックテーブルを検査するアクセスチェック手段とを備え、複数のオペレーティングシステムからアクセス可能な計算機資源を保護することを特徴とする計算機システム。

【請求項2】 オペレーティングシステム起動時に当該オペレーティングシステムの実行権限を指定するOS実行権限指定インタフェースを備えることを特徴とする請求項1記載の計算機システム。

【請求項3】 すでにロードされているオペレーティングシステムに対して、そのオペレーティングシステムの実行権限レベルを指定するOS実行権限指定インタフェースと、オペレーティングシステムの実行権限を変更するOS実行権限設定手段とをさらに備えることを特徴とする請求項1または2記載の計算機システム。

【請求項4】 オペレーティングシステムがアクセス可能な計算機資源をアクセスチェックテーブルに登録するアクセス情報設定手段を有し、オペレーティングシステム毎の計算機資源アクセス可否情報を登録することを特徴とする請求項1、2または3記載の計算機システム。

【発明の詳細な説明】

【0001】

【発明の属する技術分野】本発明は、計算機システムに係り、特に、1台の計算機上で複数のオペレーティングシステム(OS)を稼動させるマルチOS動作環境におけるOS間資源保護を図った計算機システムに関する。

【0002】

【従来の技術】通常の計算機システムは、計算機上に1つのOSを動作させ、そのOSが計算機のプロセッサ、メモリ、及び、二次記憶装置等の計算機資源を管理し、計算機が効率よく動作できるように資源スケジュールを実施している。そして、OSのコード及びデータへのアクセスは、各プログラムに与えられた実行権限により制御されている。例えば、インテルアーキテクチャ(インテル・アーキテクチャ・ソフトウェア・デベロッパーズ・マニュアル参照)の計算機は、計算機上で動作するプログラムに対して、4段階の実行権限を与えることができ、これにより、上位レベルの実行権限を持つプログラ

ムのコードあるいはデータ領域を、下位レベルの実行権限を持つプログラムから保護することができる。

【0003】また、OSには、様々な種類があり、バッチ処理に優れるものや、TSS(Time sharing System)に優れているもの、GUI(Graphical User Interface)に優れているものなどが存在している。

【0004】一方、このような様々なOSを1台の計算機で同時に実行したいというニーズがある。例えば、大型計算機においては、実際に業務に伴うオンライン処理を実行するOSと、開発用のOSとを1台の計算機で同時に稼動させたいという要求がある。また、GUIに優れているOSと、実時間性に優れているOSとを同時に稼動させたい等の要求もある。

【0005】しかし、個々のOSは、単独で計算機資源管理を実施することを仮定しており、複数のOSを共存させるためには、何らかの機構が必要である。

【0006】1台の計算機で複数のOSを動作させる機構に関する従来技術として、例えば、「OSシリーズ第11巻VM、岡崎世雄他著、共立出版株式会社発行」に記載された大型計算機で実現されている仮想計算機方式が知られている。この従来技術による仮想計算機方式は、仮想計算機制御プログラムが全ハードウェア資源を占有して管理し、それを仮想化して仮想計算機を構成するものである。そして、この仮想計算機を構成する制御部は、物理メモリ、入出力機器装置、外部割込み等を仮想化する。しかし、この仮想計算機方式は、計算機資源を完全に仮想化して分割しようとするため、仮想計算機を構成する制御部が複雑となり、また、仮想計算機の高性能化のために、特別なプロセッサ機能やマイクロコード等のハードウェア等の特殊なハードウェア機構が必要であるという問題点を有している。

【0007】また、1台の計算機で複数のOSのインターフェースを提供する方式として、マイクロカーネル方式が知られている。マイクロカーネル方式は、マイクロカーネルの上に、ユーザに提供するOS機能のインターフェースを提供するOSサーバを構築し、ユーザにそのサーバを経由して計算機資源を利用させるものである。この従来技術は、OS毎のサーバを用意することにより、ユーザに様々なOS環境を提供することができる。しかし、このマイクロカーネル方式は、OSサーバをマイクロカーネルに合わせて新規に構築する必要があり、多くの場合、現在あるOSをマイクロカーネル上で動作するように変更することになるが、スケジューリング、メモリ管理等のカーネルの中核部分も変更することになり、変更箇所が多く、また、変更箇所がOSの中核部分に及ぶため変更作業が複雑であるという問題点を有している。

【0008】前述した仮想計算機方式及びマイクロカーネル方式の問題点を解決ことのできる他の従来技術として、例えば、特開平11-149385号公報等に記載

された技術が知られている。この従来技術は、第1のOSが、第2のOSが必要とする物理メモリ、外部デバイス等の計算機資源を予約し、どちらのOSからも独立した管理プログラムが外部割り込みを横取りして、割り込み要因により、どのOSの割り込みハンドラを起動すべきかを決定し、OSの実行状態により割り込みハンドラを起動するタイミングを決定して、それに基づいて各OSの割り込みハンドラを起動することにより、2つのOSを1台の計算機で動作させるというものである。

【0009】

【発明が解決しようとする課題】前述した特開平11-149385号公報記載の従来技術は、次のような問題点を有している。

【0010】前述したように、通常のOSは、単独で計算機資源の管理を実施することを前提として構築されているおり、OSのコード及びデータ領域を保護するために、計算機の保護機構を使用し、OSに最高レベルの実行権限を与え、ユーザプログラムには、OSよりも低い実行権限を与えて動作させることにより、OSをユーザプログラムから保護している。このようなOSを特開平11-149385号公報記載の方法で、1台の計算機上に複数共存させ、各OSに最高レベルの実行権限を与えた場合、この従来技術は、特定のOSが使用する計算機資源を他のOSから隠蔽するとはいえ、ハードウェア的にアクセスを制限することができず、障害によっては、特定のOSの障害が共存している他のOSの動作に影響を与える場合があるという問題点を生じさせる。特に、現用系とテスト系とを共存させた場合、通常、テスト系には障害要因が潜んでいる可能性が現行系よりも大きい場合、テスト系の障害が現用系にも影響することになり、この点が大きな問題となる。

【0011】前述したように、1台の計算機上で共存している複数のOS間の保護を実現するためには、特別なハードウェア保護機構が必要となる。また、仮想計算機方式で複数OSを共存させる場合、高性能化のための特別なハードウェア機構を持っていない計算機は、共存しているOSすべてに関してソフトウェア的なエミュレーション処理が必要となり、性能的に問題となる。

【0012】本発明の第1の目的は、前述した従来技術の問題点を解決し、高性能化のための特別なハードウェアなしに複数のOSを共存させることが可能な計算機環境において、特別なハードウェアによるOS間資源保護機構なしにOS間の資源保護を実現することのできる計算機システムを提供することにある。

【0013】また、本発明の第2の目的は、高性能化のための特別なハードウェアなしに複数のOSを共存させることが可能な計算機環境、特に、現行系とテスト系とが共存する環境において、テスト系のみにエミュレーション処理を実施させることにより、現行系の性能を落とすことなくOS間の資源保護を行うことができる計算機

システムを提供することにある。

【0014】

【課題を解決するための手段】本発明によれば前記目的は、計算機資源のアクセス権を制限可能な実行権限レベルを複数有し、1つの計算機上で、複数のオペレーティングシステムが独自のアドレス空間と、オペレーティングシステム間共通のアドレス空間とを有し、プロセッサを複数のオペレーティングシステムにより共有して動作する計算機システムにおいて、オペレーティングシステムが実施する処理を代替するエミュレーション手段と、オペレーティングシステムの実行権限レベルを変更するOS実行権限設定手段と、オペレーティングシステムの計算機資源へのアクセス可否情報が設定されているアクセスチェックテーブルと、オペレーティングシステムが計算機資源にアクセスしようとするときにアクセスチェックテーブルを検査するアクセスチェック手段とを備え、複数のオペレーティングシステムからアクセス可能な計算機資源を保護することにより達成される。

【0015】また、本発明によれば、前述において、オペレーティングシステム起動時に当該オペレーティングシステムの実行権限を指定するOS実行権限指定インタフェースを備えることにより、すでにロードされているオペレーティングシステムに対して、そのオペレーティングシステムの実行権限レベルを指定するOS実行権限指定インタフェースと、オペレーティングシステムの実行権限を変更するOS実行権限設定手段とをさらに備えることにより、また、オペレーティングシステムがアクセス可能な計算機資源をアクセスチェックテーブルに登録するアクセス情報設定手段を有し、オペレーティングシステム毎の計算機資源アクセス可否情報を登録することにより達成される。

【0016】本発明は、前述の構成を備えることにより、OS間で資源を保護するために、通常、最高の実行権限を有して動作するOSの実行権限レベルを下げた動作させ、保護すべき資源にアクセスした場合に例外が発生するようにプロセッサを設定することができ、これにより、実行権限を下げた動作させているOSが、保護すべき資源にアクセスしようとしたことをプロセッサに通知することができる。

【0017】また、本発明は、例外が通知された場合のエミュレーション処理を登録するために、OS毎の割り込みテーブルが設置されており、このテーブルに、実行権限を下げた動作させるOSの割り込みテーブルに例外が通知された場合に実行すべきエミュレーション処理を登録する。これにより、実行権限を下げた動作させているOSが保護すべき資源にアクセスする前に、本当にアクセスして良いか否かをチェックしてからアクセスをエミュレーションすることにより、保護すべき資源へのアクセスを制限することができる。

【0018】また、本発明は、特定のOSがアクセスで

きる資源を登録するアクセスチェックテーブル登録手段を設けることにより、そのOSがアクセスする資源のアクセス情報を登録しておくことができる。

【0019】さらに、本発明は、OSの起動時あるいは動作時に、そのOSに与える実行権限レベルを指定する実行権限設定手段を設けているので、共存する特定のOSに対して、実行権限レベルを指定することができる。

【0020】さらに、本発明は、実行権限を下げて動作するOSのためのエミュレーション処理を登録するエミュレーション処理を登録することができ、これにより、実行権限を下げたOSに対するエミュレーション処理を設定することができる。

【0021】

【発明の実施の形態】以下、本発明による計算機システムの一実施形態を図面により詳細に説明する。

【0022】図1は本発明の実施形態による計算機システムの構成を示すブロック図であり、まず、本発明の実施形態による計算機システムの構成の概要を説明する。図1において、100は計算機、101はプロセッサ、102は演算装置、103はコードセグメントレジスタ、104はデータセグメントレジスタ、105はセグメントテーブルレジスタ、106は割り込みテーブルレジスタ、107はページテーブルレジスタ、108はアドレス変換装置、109は主記憶装置、110は主記憶装置の共通領域、111は第2OSの領域、112は第1OSの領域、113は割り込みコントローラ、114はROM、115は割り込みバス、116はバス、117は割り込み信号線、118はキーボード、119はディスプレイ、120は磁気ディスク装置、121、122は外部機器、130はクロック割り込み生成器である。

【0023】計算機100は、図1に示すように、プロセッサ101、主記憶装置109、バス116、割り込み信号線117、割り込みコントローラ113、ブート手順を格納している記憶装置（ROM）114、割り込みバス115、クロック割り込み生成器130を備えて構成されている。このように構成される計算機100において、割り込み信号線117は、外部入出力機器118～122と割り込みコントローラ113とを接続している。割り込みコントローラ113は、外部機器が割り込みを発生すると、割り込み信号線117を経由して信号を受け取り、割り込みコントローラ113がこの信号を数値化して、割り込みバス115を介してプロセッサ101に渡す。また、クロック割り込み生成器130は、周期的な割り込みを発生する。

【0024】プロセッサ101は、演算装置102、コードセグメントレジスタ103、データセグメントレジスタ104、セグメントテーブルレジスタ105、割り込みテーブルレジスタ106、ページテーブルレジスタ107、アドレス変換装置108を備えて構成されてい

る。

【0025】セグメントテーブルレジスタ105は、セグメントテーブル127の仮想アドレスを格納している。セグメントテーブル127の詳細は後述するが、プログラムが動作するときに使用するメモリ領域の範囲に関する情報を格納している。

【0026】コードセグメントレジスタ103は、プログラムが動作するときに使用される命令コードを格納している領域の情報を有するセグメントテーブル内のエントリの番号及び命令を実行するときの実行権限レベルを格納している。

【0027】データセグメントレジスタ104は、プログラムが動作するときに使用されるデータを格納している領域の情報を有するセグメントテーブル内エントリの番号及び当該領域をアクセスするための命令を実行するときに必要な実行権限レベルを格納している。

【0028】計算機100は、実行権限レベル0～3の4つの実行権限レベルを有しており、その権限は、レベル0が最上位の実行権限レベルであり、レベル1、レベル2、レベル3の順に実行権限が低くなる。すなわち、命令実行の観点で実行権限レベルを見た場合、実行権限レベル0は、計算機全体の制御をするシステム操作命令を実行するための命令を実行するための権限レベルであり、プロセッサが提供している命令セットのすべてを実行することができる権限レベルである。そして、その他の実行権限レベルは、システム操作命令以外の命令を実行する権限レベルである。また、メモリアクセスに関して見れば、実行権限レベル3のプログラムは、アクセスのために必要な実行権限レベルが3のデータ領域に対してのみアクセス可能であり、実行権限レベル2のプログラムは、アクセスのための権限レベルが、2と3のデータ領域にアクセスすることが可能であり、実行権限レベル1のプログラムは、アクセスのための権限レベルが、1と2と3のデータ領域にアクセス可能であり、実行権限レベル0のプログラムは、アクセスのためのレベルがどのレベルであってもデータ領域にアクセス可能である。

【0029】割り込みテーブルレジスタ106は、割り込みテーブル123の仮想アドレスを指し示している。割り込みテーブル123の詳細については後述するが、割り込み番号毎の割り込みハンドラの開始アドレスを記憶している。割り込みが発生すると、プロセッサ101は、割り込みコントローラ113から数値化された割り込み番号を受け取り、この番号をインデックスとして割り込みテーブル123により割り込みハンドラアドレスを取得し、割り込みハンドラに制御を渡す。また、割り込みテーブル123は、計算機100に接続されている機器からの割り込みだけでなく、プロセッサ内で発生した例外についても、例外が発生したときの処理を登録している。

【0030】ページテーブルレジスタ107は、ページテーブル128を指し示している。ページテーブルレジスタは、ページテーブル128の物理アドレスを格納している。

【0031】アドレス変換装置108は、演算装置102が要求する命令アドレスあるいはオペランドが格納されているアドレスを受け取り、ページテーブルレジスタ107が指しているページテーブル128の内容に基づいて、仮想アドレス—物理アドレス変換を実施する。さらに、アドレス変換装置108は、仮想アドレス—物理

アドレス変換の前に、演算装置が要求するアドレスへのアクセス可否のチェックを行い、アクセス不可の場合、例外としてプロセッサに通知する。

【0032】計算機100には、外部入出力装置として、キーボード118、ディスプレイ119、磁気ディスク装置120、その他の外部機器121及び122が接続されている。ディスプレイ119を除く機器は、割り込み信号線117により割り込みコントローラ113に接続されている。

【0033】次に、計算機100上で第1OSと第2OSとの2つのOSが動作しているものとして、主記憶装置109の内容について説明する。なお、ここでは、計算機100を起動すると、第1OSが起動するように設定されており、外部機器121、122は第2OSにより管理される機器であるとする。

【0034】第1OSは、計算機100の起動時に、その他のOS用、この場合、第2OS用に物理メモリ領域を予約する。すなわち、第1OSが、第2OS用に予約された物理メモリ領域を利用できないように物理メモリ領域を確保する。図1には、この予約した領域に第2OSがロードされている様子を示している。また、第1OSの初期化課程において、第1OSから外部機器121、122の利用する割り込み番号や入出力アドレスが、すでに利用済みであるとして予約される。

【0035】第1OSは、全てのOSから参照可能な共通領域110を持つ。その共通領域110には、割り込みテーブル123、アクセスチェックテーブル129、OS実行権限設定手段126、割り込みハンドラ125、エミュレーション手段124等が格納されている。また、第1OS領域112及び第2OS領域111には、それぞれ、各OSが使用する仮想空間とその仮想空間を構成する物理メモリ領域のマッピング情報とを有するページテーブル128、及び、構成した仮想空間をアクセスするための実行権限レベル情報を有するセグメントテーブル127が存在する。なお、図1におけるセグメントテーブル127、ページテーブル128は、第2OSのテーブルであり、第1OSのセグメントテーブル、ページテーブルも第1OSの領域112に存在するが図の簡略化のため記載を省略している。

【0036】図2はセグメントテーブルの構成を説明す

る図であり、次に、図2を参照して、本発明の実施形態におけるセグメントテーブル127の構成について説明する。

【0037】セグメントテーブル127は、仮想空間内の特定の領域へアクセスするための実行権限レベル及びその領域の属性を記録したテーブルであり、エントリ番号201が付与された複数のエントリを有している。

【0038】図2において、ベースアドレス202は仮想空間内の特定領域の開始アドレスを示し、リミット203は領域長を示し、実行権限レベル204は当該領域へのアクセスに必要な実行権限レベルを示している。タイプ205はその領域に格納する情報のタイプ、すなわち、命令コードが格納されるのか、データが格納されるのかを示すタイプを示しており、図2に示す例では、命令コード領域の場合にC、データ領域の場合にDとして表記している。206は、そのエントリが有効か否かを示している。

【0039】計算機100は、プロセッサがプログラムを実行する場合、そのプログラムを実行する前に、コードセグメントレジスタ103に格納されているプロセッサが実行するプログラムコードCが格納されている領域のセグメント情報（セグメントテーブル127内に記録されている情報）を示すセグメント内インデックスをロードし、そのプログラムが参照するデータ領域のセグメント情報を示すセグメント内インデックスをロードする。コードセグメントレジスタ103にローディングされたセグメント情報の実行権限レベルが、そのプログラムを実行するときの実行権限レベルとなる。計算機100は、プロセッサ101がデータにアクセスする前に、そのプログラムが有する実行権限レベルと、アクセスしようとするデータ領域をアクセスするための実行権限レベルとを比較し、現在プログラムが有する実行権限レベルがアクセスしようとするデータ領域をアクセスするために必要な実行権限レベルと同等かそれ以上の場合にアクセスを許可する。アクセスのために必要な実行権限レベルより低い実行権限レベルの場合、アクセス例外として、プロセッサに通知する。

【0040】また、計算機100は、実行権限レベル0以外の実行権限を有するプログラムが、システムを制御するシステム操作命令（例えば、データセグメントレジスタ104を操作する命令等）を実行しようとした場合、命令実行前に、プログラムの実行権限レベルをチェックし、実行権限レベルが0以外の場合、実行権限例外として、プロセッサに通知する。この機構により、計算機100は、実行権限レベルの低いプログラムが、高い実行権限を有するプログラムが使用する領域を破壊することを防止する。

【0041】図3はページテーブルの構成を説明する図であり、次に、図3を参照して、本発明の実施形態におけるページテーブル128の構成について説明する。

【0042】ページテーブル128は、プロセッサ101の仮想アドレス空間の仮想ページ毎に、それぞれの仮想ページを記述する複数のエントリを有している。それぞれのエントリは、仮想ページ番号301と、有効ビット302と、物理ページ番号303とにより構成されている。有効ビット302は、その仮想ページに対応する物理ページが割り当てられているか否か、すなわち、仮想-物理アドレス変換が可能か否かを示している。例えば、図3におけるページテーブル128の仮想ページ番号2に対応する有効ビット302は、“0”であり、“1”にセットされていないため、仮想ページ番号2に対応する物理ページは存在しないことを示している。そして、有効ビット302が“1”に設定されていない仮想ページにアクセスが発生すると、プロセッサはページフォルトを発生する。

【0043】物理ページ番号303は、仮想ページに対応する物理ページ番号を格納している。アドレス変換装置108は、ページテーブルレジスタ107の指しているページテーブル128を参照して、演算装置102が生成する仮想アドレスを物理アドレスに変換する。プロセッサ101は、アドレス変換により得られた物理アドレスにより主記憶装置109を参照する。

【0044】ページテーブルを切り替えることにより、主記憶109の中に独立した空間を構築することができ、第1OSの空間及び第2OSの空間を別々に構築することが可能である。また、主記憶109内の共通領域110については、両OSのページテーブルの共通領域に対応する部分に、同じ物理ページをマップするよう設定しておくことにより、共通領域を実現することができる。

【0045】図4は割り込みテーブルの構成を説明する図であり、次に、図4を参照して、本発明の実施形態における割り込みテーブル123の構成について説明する。

【0046】割り込みテーブル123は、プロセッサ101が受け取る割り込み番号毎401の割り込みハンドラの仮想アドレス402を記憶している。プロセッサ101は、割り込みを受け取ると、受け取った割り込み番号に対応する割り込みハンドラアドレスを、割り込みテーブルレジスタ106が指す割り込みテーブル123から取得し、そのアドレスに制御を移すことにより割り込み処理を開始する。ここでいう割り込みとは、計算機100に接続されている外部機器からのデバイス割り込み及び命令実行時にアクセス例外等でプロセッサに通知される例外を意味する。

【0047】図5はアクセスチェックテーブルの構成を説明する図であり、次に、図5を参照して、本発明の実施形態におけるアクセスチェックテーブル129の構成について説明する。

【0048】アクセスチェックテーブル129は、OS

がアクセス可能である領域の先頭アドレスを示すアクセス許可領域先頭アドレス502と、アクセス可能領域のサイズを示すアクセス許可領域長503と、そのエントリが有効であるか否かを示す有効ビット504とから構成されている。なお、これらの502、503、504をアクセスチェック情報と呼ぶ。また、アクセスチェックテーブル500は、1つ以上のエントリから構成されており、各エントリには、エントリ番号として、インデックス501が付与されている。アクセス許可領域長503は、アクセス許可領域の大きさをバイト単位で記録している。

【0049】このアクセスチェックテーブル129を用いることにより、特定のOSが資源にアクセスしようとした場合に、アクセスして良いか否かを判定することができる。また、アクセスチェックテーブル129の内容については、予め決まっている内容に固定的に設定しても構わないが、共存するOSからアクセスチェックテーブルの内容を設定するためのアクセスチェックテーブル設定インタフェースを有するアクセスチェックテーブル設定手段を設け、計算機上で共存しているOSからアクセスチェック情報を登録する形態にしてもよい。

【0050】図6は主記憶装置上に第1OS及び第2OSがローディングされてこれらのOSが1台の計算機上に共存している状態を示す図、図7は第2OSが通常使用するセグメントテーブルの構成を示す図、図8は本発明の実施形態により使用する第2OSのセグメントテーブルの構成を示す図、図9はセグメントテーブルの内容の変更により、セグメントテーブルに格納されているセグメント情報で示される仮想空間の実行権限が変更されることを説明する図、図10はアクセス例外が発生したときのエミュレーション処理の動作を説明するフローチャート、図11は実行権限例外が発生したときのエミュレーション処理の動作を説明するフローチャート、図12は第2OSが構成する仮想空間の構成を示す図、図13はOSの実行権限レベルを変更するOS実行権限設定処理について説明する図であり、次に、図6～図13を参照して、本発明の実施形態により、第1OSと第2OSとを共存させ、両OSから共用する資源に対して第2OSのアクセスを制限するOS間資源保護を実現することについて説明する。

【0051】図6に示すように、主記憶109には、第1OS及び第2OSで共通に使用する共通領域110と、第2OS固有の領域111と、第1OS固有の領域112とが形成される。第2OSの領域111には、第2OSの命令コードが格納されている第2OS命令コード領域709と第2OSが参照するデータがされている第2OSデータ領域710とが設けられ、データ領域710内には、第2OS用のセグメントテーブル711と、ページテーブル712とが配置される。

【0052】第1OS領域112には、第1OSの命令

コードが格納されている第1 OS 命令コード領域713と第1 OS が参照するデータがされている第1 OS データ領域714とが設けられ、データ領域710内には、第2 OS 用のセグメントテーブル715と、ページテーブル716とが配置される。

【0053】また、共通領域110には、図1により説明したように、割り込みテーブル123、割り込みハンドラ125、OS 実行権限設定手段126が配置されると共に、さらに、第2 OS エミュレーション手段707と、第2 OS アクセスチェックテーブル708とが配置されている。割り込みテーブル123の各エントリには、割り込みが発生した場合に実行する処理のスタートアドレスとして、割り込みハンドラ125、及び、第2 OS エミュレーション手段707を実行するためのアドレスが登録されている。

【0054】第2 OS が構成する仮想空間は、図12に示すように、ユーザプログラム命令コード領域1301、ユーザプログラムデータ領域1302、OS 命令コード領域1303、OS データ領域1304、共通領域1305から構成される。そして、図12に示すような仮想空間を構成したとき、第2 OS が通常使用するセグメントテーブル711は図7に示すように構成される。

【0055】図7に示す第2 OS が通常使用するセグメントテーブル711において、エントリ810は、OS 命令コード領域1303のセグメント情報を格納しており、エントリ811は、OS データ領域1304の、エントリ812、813は、共通領域1305の、エントリ814は、ユーザプログラム命令コード領域1301の、エントリ815は、ユーザプログラムデータ領域1302の各セグメント情報を格納している。

【0056】本発明の実施形態は、第2 OS のセグメントテーブル711を図8に示すように変更して使用する。図8に示すテーブルの図7に示すテーブルとの相違は、図7に示すテーブルのエントリ810に対応する図8のテーブルのエントリ910の実行権限レベル804の内容を“1”にしていることである。エントリ910の実行権限レベルを“0”から“1”に変更することにより、第2 OS がOS 領域、すなわち、エントリ811、812、813で記述されている領域をアクセスしようとした場合、プロセッサに対して、アクセス例外、あるいは、実行権限例外として通知されることになる。なお、セグメントテーブル内の実行権限レベル804を変更するのは、図1及び図6に示しているOS 実行権限設定手段126である。

【0057】セグメントテーブルの内容を図7に示す内容から図8に示す内容に変更することにより、セグメントテーブルに格納されているセグメント情報で示される仮想空間の実行権限は、図9に示すように変更されることになる。

【0058】次に、前述したように、第2 OS の実行権

限を変更した場合に発生するアクセス例外、及び、実行権限例外発生時の第2 OS のエミュレート処理について、図10、図11に示すフローを参照して説明する。まず、図10に示すフローを参照して、はアクセス例外が発生したときのエミュレーション処理について説明する。なお、エミュレート処理は、図6に示す第2 OS エミュレーション手段により実施される。

【0059】(1) まず、例外が発生した命令のデコード処理を実施する。そして、このデコード処理において、例外が発生した命令の命令語、参照レジスタ、参照アドレス等を取得する(ステップ1101)。

【0060】(2) 次に、例外発生命令を実行しようとしたときの命令実行権限レベルを、コードセグメントレジスタあるいはセグメントテーブルから取得する。その後、ステップ1101の処理で取得した参照アドレスが指す領域を含むセグメントに対するデータ参照権限レベルを、データセグメントレジスタあるいはセグメントテーブルから取得する(ステップ1102、1103)。

【0061】(3) ステップ1102の処理で取得した実行権限レベルが“1”であるか否かをチェックし、実行権限レベルが“1”以外の場合、他の例外処理に制御を移す(ステップ1104、1109)。

【0062】(4) ステップ1104の判定で、実行権限レベルが“1”であった場合、その例外発生命令が実行権限を“1”に変更したセグメント内に格納されている命令であるか否かをチェックし、その命令が実行権限レベルを“1”に変更したセグメント内の命令でなかった場合、他の例外処理に制御を移す(ステップ1105、1109)。

【0063】(5) ステップ1105の判定で、その命令が実行権限レベルを“1”に変更したセグメント内の命令であった場合、その例外発生命令の参照データが、アクセスチェックテーブル708に登録されている領域内であるか否かをチェックし、その例外発生命令の参照データがアクセスチェックテーブルに登録されていない場合、他の例外処理に制御を移す(ステップ1106、1109)。

【0064】(6) ステップ1106の判定で、その例外発生命令の参照データが、アクセスチェックテーブル708に登録されている領域内であった場合、例外発生命令を実際に行うエミュレート処理を実行する(ステップ1107)。

【0065】(7) 例外処理を終了して例外発生命令の次の命令に制御を移すための復帰アドレスを設定する処理を実行する(ステップ1108)。

【0066】前述した処理において、ステップ1109へ制御が渡るのは、そのOS がアクセスする資源がアクセスチェックテーブルに登録されていない場合、あるいは、実行権限レベルを変更したことにより発生したアクセス例外ではなく、プログラムのバグ等の要因で発生し

たアクセス例外が発生した場合である。このような例外が発生した場合の例外処理がステップ1109である。

【0067】図10により説明したアクセス例外処理を実行することにより、アクセス可否をチェックすることができ、アクセスチェックテーブルに、アクセス許可資源を登録することにより、登録されていない資源へのアクセスを禁止することができる。

【0068】次に、実行権限例外が発生したときのエミュレーション処理について、図11に示すフローを参照して説明する。

【0069】(1) まず、例外が発生した命令のデコード処理を実施する。そして、このデコード処理において、例外が発生した命令の命令語、参照レジスタ、参照アドレス等を取得する(ステップ1201)。

【0070】(2) 次に、例外発生命令を実行しようとしたときの命令実行権限レベルを、コードセグメントレジスタあるいはセグメントテーブルから取得する(ステップ1202)。

【0071】(3) ステップ1202の処理で取得した実行権限レベルが“1”であるか否かをチェックし、実行権限レベルが“1”以外の場合、他の例外処理に制御を移す(ステップ1203、1207)。

【0072】(4) ステップ1203の判定で、実行権限レベルが“1”であった場合、その例外発生命令が実行権限を“1”に変更したセグメント内に格納されている命令であるか否かをチェックし、その命令が実行権限レベルを“1”に変更したセグメント内の命令でなかった場合、他の例外処理に制御を移す(ステップ1204、1207)。

【0073】(5) ステップ1204の判定で、その例外発生命令が実行権限レベルを“1”に変更したセグメント内の命令であった場合、例外発生命令を実際に実行するエミュレート処理を実行する(ステップ1205)。

【0074】(6) 例外処理を終了して例外発生命令の次の命令に制御を移すための復帰アドレスを設定する処理を実行する(ステップ1206)。

【0075】前述した処理において、ステップ1207へ制御が渡るのは、実行権限レベルを変更したことにより発生した実行権限例外ではなく、プログラムのバグ等の要因で発生した実行権限例外が発生した場合である。このような例外が発生した場合の例外処理がステップ1207である。

【0076】図10、図11により説明した処理を実行することにより、実行権限レベルを“1”に変更したプログラム(本発明の実施形態では、第2OS)は、実行権限レベルを“1”に下げて実行することができ、さらに、アクセスチェックテーブルに登録していない資源に対してアクセスを禁止することができる。

【0077】次に、図13を参照して、OSの実行権限

レベルを変更するOS実行権限設定処理について説明する。

【0078】図13には、OS実行権限設定手段1401が共通領域に配置され、OS実行権限設定手段1401が、OS実行権限指定インターフェース1402を有していることが示されている。インターフェース1402は、計算機上で共存しているOSから特定OSの実行権限を指定するためのインターフェースであり、その入力として、どのOSの実行権限を設定するかを示すOS識別子と、設定すべき実行権限レベルとを必要とする。

【0079】入力値として、OS識別子と実行権限レベルとを与えられたOS実行権限設定手段1401は、実行権限レベルを設定すべきOSのセグメントテーブル内の命令を格納する領域の情報を格納するエントリに対して、その実行権限レベルを入力されたレベルに設定し、さらに、割り込みテーブルに対して、実行権限レベルを変更することにより発生する例外に対する処理として、エミュレーション処理を実行するための命令アドレスを設定する。ここで、変更するセグメントテーブルエントリについては、予め、判っているものとする。OS実行権限設定手段1401により、1台の計算機上で共存しているOSから、特定のOSに対して、実行権限レベルを変更することが可能となる。なお、前述で説明したOS実行権限設定処理は、1台の計算機上で共存し、現在実行中のOSから、現在実行していないもう一方のOSに対して実施可能であるが、特定のOS起動時の初期化処理として、前述したインターフェースを使用することにより、実行権限を設定することが可能である。

【0080】前述した本発明の実施形態によれば、1台の計算機上で2つのOSが共存する環境において、OSの実行権限を設定するOS実行権限設定手段を設け、1つのOSの実行権限を下げた実行し、実行権限を下げたOSの処理をエミュレーションすることにより、共存するOS間で共用する計算機資源に対するアクセスを制限することができる。そして、本発明の実施形態によれば、もう一方のOSに関しては、エミュレーション処理を行わずにそのまま実行させることができ、性能に関しても、高速化のための特別ハードウェアを必要としない。特に、共存させるOSとして、通常使用する現用系OSと、テスト系OSとを共存させた場合、テスト系OSに障害要因が潜んでいる場合が一般的に多く、テスト系OSからの資源アクセスを制限することにより、テスト系OSの障害が現用系OSに波及することを防止することができる。このように、テスト系OSの処理のみエミュレーション処理を実行することにより、現用系OSに関する性能劣化も防止することができる。

【0081】前述した本発明の実施形態は、共存するOSが2つであるとして説明したが、本発明は、共存するOSが3つ以上である場合にも、第2OS、第3OSに対して、実行権限を下げることにより、同等の効果を

ることができる。

【0082】また、前述した本発明の実施形態は、第2 OSのみの実行権限を下げることにより、第2 OSからの資源アクセスを制限したが、第2 OSと同様に、第1 OSの実行権限を下げることにより、共存するOS間で共用する資源のアクセスを両OSから制限することも可能である。

【0083】また、前述した本発明の実施形態は、本発明を単一のプロセッサを有する計算機に適用したとして説明したが、本発明は、複数のプロセッサを持つ計算機 10 に対しても同様に適用することが可能である。

【0084】さらに、前述した本発明の実施形態は、OSがアクセスする資源として、主記憶装置を対象として説明したが、本発明は、インテルアーキテクチャのI/O空間のような主記憶装置以外の資源を有する計算機に対しても適用することができ、主記憶装置と同様に、主記憶装置以外の資源を同様な方法で保護することが可能となる。

【0085】

【発明の効果】以上説明したように本発明によれば、複 20 数OSを共存させる計算機環境において、高性能化のための特別なハードウェアなしに、かつ、特別なハードウェアによるOS間資源保護機構なしに、OS間の資源保護を実現することができ、特に、現行系とテスト系とが共存する環境において、テスト系のみにエミュレーション処理を実施することにより、現行系の性能を落とすことなくOS間での資源保護を図ることができる。

【図面の簡単な説明】

【図1】本発明の実施形態による計算機システムの構成を示すブロック図である。

【図2】セグメントテーブルの構成を説明する図である。

【図3】ページテーブルの構成を説明する図である。

【図4】割り込みテーブルの構成を説明する図である。

【図5】アクセスチェックテーブルの構成を説明する図である。

【図6】主記憶装置上に第1 OS及び第2 OSがローデ

ィングされてこれらのOSが1台の計算機上に共存している状態を示す図である。

【図7】第2 OSが通常使用するセグメントテーブルの構成を示す図である。

【図8】本発明の実施形態により使用する第2 OSのセグメントテーブルの構成を示す図である。

【図9】セグメントテーブルの内容の変更により、セグメントテーブルに格納されているセグメント情報で示される仮想空間の実行権限が変更されることを説明する図 である。

【図10】アクセス例外が発生したときのエミュレーション処理の動作を説明するフローチャートである。

【図11】実行権限例外が発生したときのエミュレーション処理の動作を説明するフローチャートである。

【図12】第2 OSが構成する仮想空間の構成を示す図である。

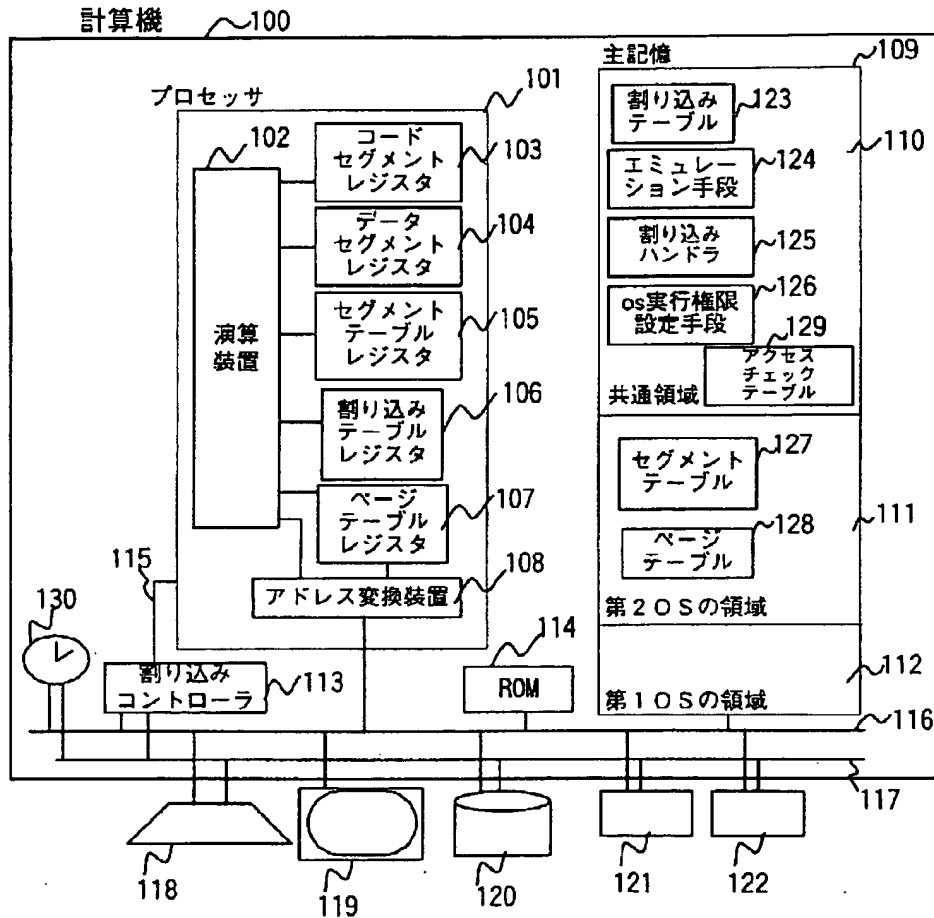
【図13】OSの実行権限レベルを変更するOS実行権限設定処理について説明する図である。

【符号の説明】

- 100 計算機
- 101 プロセッサ
- 102 演算装置
- 103 コードセグメントレジスタ
- 104 データセグメントレジスタ
- 105 セグメントテーブルレジスタ
- 106 割り込みテーブルレジスタ
- 107 ページテーブルレジスタ
- 108 アドレス変換装置
- 109 主記憶装置
- 30 110 共通領域
- 111 第2 OS領域
- 112 第1 OS領域
- 113 割り込みコントローラ
- 130 クロック生成器
- 116 バス
- 117 割り込み信号線

【図1】

図1



【図3】

図3

仮想ページ番号 128	有効ビット 301	物理ページ番号 302
0	1	0
1	0	0
2	0	0
3	1	32
4	1	64
5	1	2

【図4】

図4

割り込み番号 123	割り込みハンドラ開始アドレス 401
0	800h
1	500h
2	520h
3	670h
4	230h
5	620h

【図2】

図2

127 No	201 ベース アドレス	202 リミット	204 実行権限レベル	205 タイプ	206 有効
0	0h	0h	0	0	0
1	0h	0h	0	C	0
2	80000000h	10000000h	0	C	1
3	A0000000h	10000000h	0	D	1
4	0h	40000000h	3	C	1
5	40000000h	40000000h	3	D	1
6	0h	FFFFFFFFh	3	Q	0
⋮					
n					

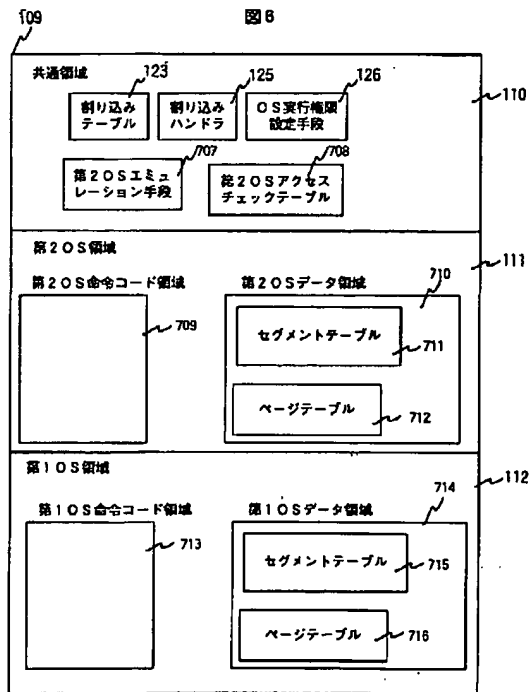
【図5】

図5

129 インデ ックス	501 アクセス許可領域 先頭アドレス	502 アクセス 許可領域長	503 有効
0	FFFFFFFF000h	FFFh	1
1	80C4FdFCh	1h	1
2	F0001008h	4h	1
3	D0065600h	FFh	1
4	FFFFFFFF000h	0h	0
5	FFFFFFFF000h	0h	0
⋮			

【図6】

図6



【図7】

図7

711 No	802 ベース アドレス	803 リミット	804 実行権限レベル	805 タイプ	806 有効
0	0h	0h	0	0	0
⋮					
10	80000000h	10000000h	0	C	1
11	90000000h	30000000h	0	D	1
⋮					
22	c0000000h	40000000h	0	C	1
23	c0000000h	40000000h	0	D	1
⋮					
51	0h	40000000h	3	C	1
52	40000000h	40000000h	3	D	1
⋮					
n					

【図8】

図8

	801	802	803	804	805	806
	No	ベース アドレス	リミット	実行権限レベル	タイプ	有効
	0	0h	0h	0	0	0
	10	80000000h	100000000h	1	C	1
	11	90000000h	30000000h	0	D	1
	22	c0000000h	40000000h	0	C	1
	23	d0000000h	40000000h	0	D	1
	51	0h	40000000h	3	C	1
	52	40000000h	40000000h	3	D	1
	n					

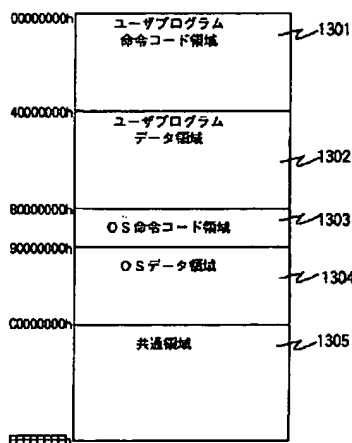
【図9】

図9

	実行権限レベル
	<変更前> <変更後>
00000000h ユーザプログラム 命令コード領域	3 → 3
40000000h ユーザプログラム データ領域	3 → 3
80000000h OS命令コード領域	0 → 1
90000000h OSデータ領域	0 → 0
c0000000h 共通領域	0 → 0
FFFFFFFFh	

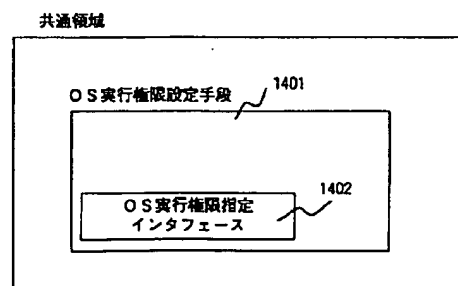
【図12】

図12



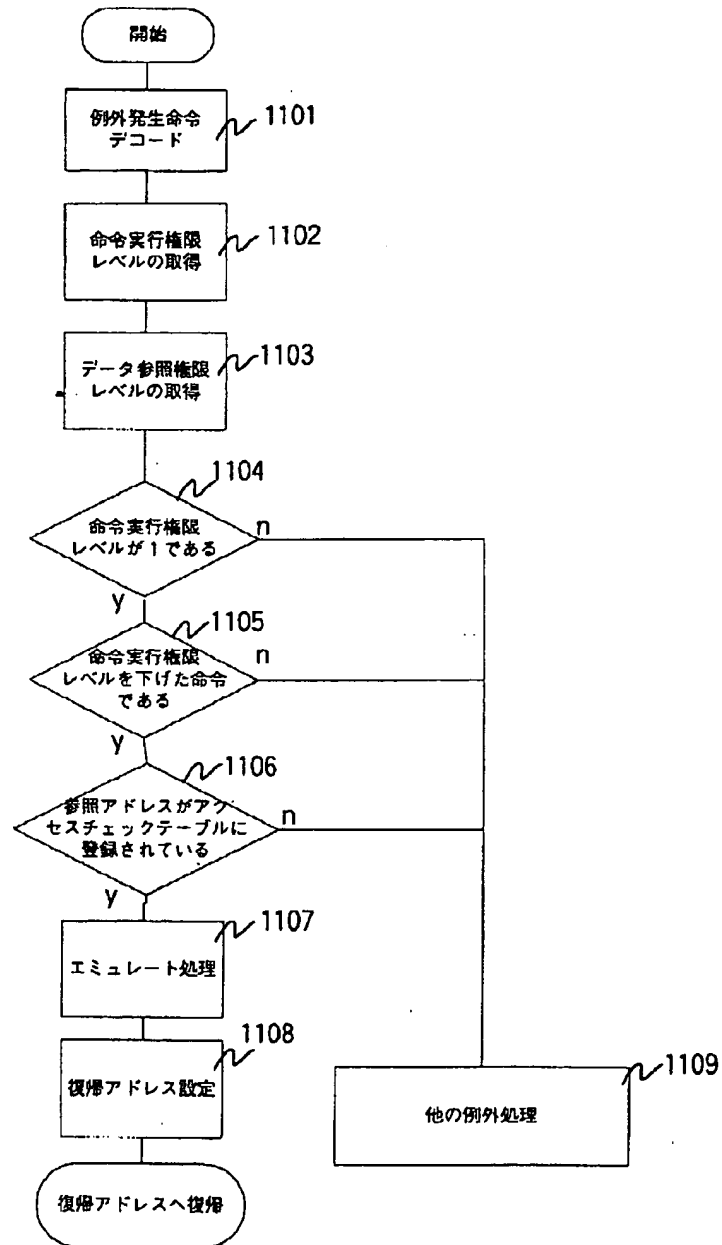
【図13】

図13



【図10】

図10



【図11】

図11

